mqdtfit: User Manual

R. M. Potvliege
Physics Department, Durham University, Durham DH1 3LE, UK

October 2023

Contents

1 About these programs

The programs forming this library are Python 3 functions for empirical multichannel quantum defect theory (MQDT) calculations of properties of atomic bound states. These functions are organised into a single module. Their purpose and the corresponding theoretical and computational methods are described in the Computer Physics Communications article accompanying this distribution. General information and important details about using this module can also be found in this article, as well as a brief description of the two example programs included in this distribution.

A more detailed description of the 14 user-facing functions currently included in this module is given in Section 2. The module also includes 26 functions not intended to be called directly by an external program. A technical description of the input parameters of each of these 40 functions and of the results they return can be found at the start of the respective code.

Revised versions of these programs and new related material will be made available from the URL https://github.com/durham-qlm/mqdtfit (a GitHub repository of the Quantum, Light and Matter Group of Durham University).

2 User-facing functions

The functions described below are intended to interface with a user-written driving program. All equation numbers and section numbers cited in this section refer to the Computer Physics Communications article accompanying this distribution.

- `initialize_eigenchannel`: This function initialises or re-initialises calculations in the eigenchannel formulation of the theory. The driving program must call either `initialize_eigenchannel` or `initialize_Kmatrix` before any other function of the module. This call defines the fundamental parameters of the MQDT model considered, i.e.:
  - the number of channels;
  - the ionization limit of each dissociation channel;
  - the orbital angular momentum quantum number of the outer electron for each of these channels;
  - the values of I_s and of $\tilde{R}$;
  - the experimental energies and experimental errors to be used in the calculation (as a single array of experimental data);
  - the number of rotation matrices and the p and q indices of each rotation;
  - the $[U_{i\bar{\alpha}}]$ matrix (this could be a unit matrix if no transformation between coupling schemes would be necessary);
  - the channel indices of the j- and k-channels;
  - whether each of the $\mu_\alpha^{(1)}$'s is a static or a fitting parameter;
  - whether the eigen quantum defects should be calculated as per Eq. (30) or per Eq. (31) (the choice between these two options is made by setting the variable `option_Ecorrection` to, respectively, `"E"` or `"A"`);
  - and whether the theoretical energies should be calculated through a direct search for the solutions of Eq. (19) or through a search for the minima of $\Xi^2(\nu_j)$ (the choice between these two options is made by setting the variable `calculation_method` to, respectively, `"zerodeterm"` or `"Xi2min"`).
- `initialize_Kmatrix`: Similar to `initialize_eigenchannel` but for the initialisation or re-initialisation of calculations in the K-matrix formulation of the theory. Apart from those specific to only one of these two formulations, the arguments are the same as for `initialize_eigenchannel`: instead of information about rotation matrices and eigen quantum defects, `initialize_Kmatrix` expects information about whether each of the $K_{ij}^{(0)}$'s and $K_{ii}^{(1)}$'s is a static or or a fitting parameter, and whether the K_{ii}'s should be calculated as per Eq. (33) or per Eq. (34) (the choice between these two options is made by setting the variable `option_Ecorrection` to, respectively, `"E"` or `"A"`, as for `initialize_eigenchannel`).

- `list_Zcoeffsandchannelfractions`: Writes out tables of mixing coefficients and channel fractions to the standard output stream, and optionally also outputs those to a file, given the parameters of an MQDT model. There are two mandatory arguments:

  - the parameters of the model as a 1D-array;
  - a set of experimental data (the same experimental energies and experimental errors as previously passed to the module through one of the `initialize` functions, or a subset of those).

  The mixing coefficients and channel fractions are calculated and printed only for the theoretical energies closest to the experimental energies passed through the list of arguments (the experimental errors are not used). This function also has optional arguments:

  - `recouple`: A boolean variable. If `recouple` is `True`, the mixing coefficients and channel fractions are first transformed from the dissociation channel coupling scheme to the intermediate coupling scheme (the "$\bar{\alpha}$ scheme") before being written out. No transformation takes place if `recouple` is `False` or is not specified.
  - `with_dbydE`: A boolean variable. If `with_dbydE` is `True`, the mixing coefficients are normalized as per Eq. (12) or Eq. (32). If `with_dbydE` is `False` or is not specified, the mixing coefficients are normalized as per Eq. (15).
  - `file`: If specified, the name of the file to which the tables of mixing coefficients and channel fractions should be written in addition to being written to the standard output stream.

- `LuFano_plot`: Displays a Lu-Fano plot, as described in Section 3.4 of the article, and optionally saves this plot to a file in pdf format. The theoretical values of ν_k are represented by individual dots if $N_k > 1$ or by a solid curve if $N_k = 1$. The experimental values are represented by circles. As is customary, $-\nu_k$ is plotted, not ν_k. This function has six mandatory arguments:

  - the parameters of the model as a 1D-array;
  - the smallest value of ν_j to be considered;
  - the largest value of ν_j to be considered;
  - the step in ν_j used in the plot;
  - the smallest value of $\nu_k \bmod 1$ to be considered;
  - the largest value of $\nu_k \bmod 1$ to be considered.

  This function also has an optional argument:

  - `file`: If specified, the name of the file to which the plot should be saved.

- `mqdtparams_eigenchannel`: Given a set of values of $\mu_\alpha^{(0)}$ and $\mu_\alpha^{(1)}$ and a set of rotation angles, returns a single 1D array containing these parameters stored in the format expected by other functions of the modules.

- `mqdtparams_Kmatrx`: Given a set of values of $K_{ij}^{(0)}$ and $K_{ii}^{(1)}$, returns a single 1D array containing these parameters stored in the format expected by other functions of the modules. This function makes no use of the values of $K_{ij}^{(0)}$ for $i > j$.

- `optimizeparams`: Optimizes the parameters of an MQDT model by χ^2-fitting as described in Section 3.3 of the article and returns them as a 1D array. This function has a single mandatory argument, namely a 1D array containing the initial values of these parameters. It also has five optional arguments:

  – `monitor`: A boolean variable. Information making it possible to monitor the progress of the optimization process is written out if `monitor` is `True`.

  – `nonzdelt`: The same `nonzdelt` parameter controlling the shape of the initial simplex as in the function `_minimize_neldermead` of the `scipy.optimize` library.

  – `zdelt`: The same `zdelt` parameter controlling the shape of the initial simplex as in the function `_minimize_neldermead`.

  – `maxfun`: The maximum number of function evaluations. The default value of this parameter is 3,000.

  – `maxiter`: The maximum number of iterations. The default value of this parameter is 3,000.

- `plot_channelfractions`: Displays a plot of channel fractions and optionally saves this plot to a file in pdf format. The following arguments are mandatory:

  – the parameters of the model as a 1D array;

  – the smallest energy to be considered;

  – the largest energy to be considered;

  – the upper limit of the range of channel fractions appearing in the plot (this value controls the scale of the vertical axis);

  – a set of experimental data from which the experimental energies to be included in the plot should be extracted (the same experimental energies and errors as the set previously passed to the module through one of the `initialize` functions, or a subset of this).

  – the value(s) of the index i or the index $\bar{\alpha}$ for which the channel fractions should be plotted.

  This function also has three optional arguments:

  - `recouple`: As for `list_Zcoeffsandchannelfractions`.
  - `with_dbydE`: As for `list_Zcoeffsandchannelfractions`.
  - `file`: If specified, the name of the file to which the plot should be saved.

  The experimental energies are represented by vertical dashed lines (as in the rest of this software, energies are expressed as wave numbers specified in cm^{-1}). Channel fractions are represented by markers and plotted for up to four channels: open circles are used for the first channel specified in the list of arguments, filled circles for the second, open squares for the third and filled squares for the fourth.

- `print_channelparams`: Writes the general details of the model to the standard output stream, i.e., the value of I_s, the value of $\tilde{R}$, and the ionization limit of each channel. The value of l_i for each channel is also written out if the optional argument `printlvalues` is `True`. The transformation matrix $[U_{i\bar{\alpha}}]$ is also written out if the optional argument `printUialphabarmatrx` is `True`. This information is also written to a file if the optional argument `file` giving the name of that file is specified. This function does not take any other argument.

- `print_chi2`: Prints out the χ^2 and reduced χ^2 statistics quantifying the goodness of fit of a given MQDT model with the set of experimental energy levels previously passed to the module through one of the `initialize` functions. The parameters of the model must be passed to this function through its list of arguments as a single 1D array. This function has two optional arguments:

  - `moreinfo`: A boolean variable. The number of data points, the number of fitting parameters and the number of degrees of freedom are also printed if `moreinfo` is `True` (the number of fitting parameters is the number of parameters not declared as being static at initialisation, and the number of degrees of freedom is the difference between the number of data points and the number of fitting parameters).
  - `file`: If specified, the name of a file to which these results should also be written in addition to being written to the standard output stream.

- `print_energies`: Prints out experimental energies and the corresponding calculated energies for a given MQDT model. This function has two mandatory arguments, namely the parameters of the model formatted as a single 1D array, and the set of experimental data to be included in the table (these should be identical to the set previously passed to the module through one of the `initialize` functions or be a subset of this). This function also has one optional argument:

  - `file`: If specified, the name of a file to which the results should also be written in addition to being written to the standard output stream.

- `print_mqdtparams`: Writes the parameters of one or several models to the output streams, i.e., the $K_{ij}^{(0)}$'s and $K_{ii}^{(1)}$'s for calculations in the K-matrix formulation of MQDT or the $\mu_\alpha^{(0)}$'s, $\mu_\alpha^{(1)}$'s and θ_m's for the calculations in the eigenchannel formulation. At least one set of MQDT parameters must be passed to this function. Several sets may be passed in the same call, each as an individual 1D array. The parameters are tabulated in as many columns as parameter sets have been passed to the function. This information is also written to a file if the optional argument `file` giving the name of that file is specified. This function does not take any other argument.

- `reset_calculation_method`: This function resets the method used for calculating the energies. It has only one argument, namely a character string which must be either `"Xi2min"` or `"zerodeterm"` (see the description of the function `initialize_eigenchannel` for further information about this choice).

- `set_searchintervals`: Sets the search intervals used in the calculation of energies as described in Section 3.2 of the article. This function has one mandatory argument, namely a single character string determining whether the intervals should be adapted to the separation between the experimental values of ν_j as per Eq. (40), or should not depend on that separation and be calculated instead as per Eq. (39). These two possibilities correspond respectively to the values `"R"` and `"A"` of this argument. This function also has two optional arguments:
  - `deltanuj`: The value of $\delta\nu$ in calculations under the option `"A"`. The default value of $\delta\nu$ is 0.01.
  - `factor`: The value of the factor α in calculations under the option `"R"`. The default value of α is 1/3.